Primitivas geométricas en SDL

Para dibujar figuras geométricas usaremos `SDL2_gfx`[1] de *ferzkopp*.

Todas las primitivas reciben un `renderer` donde dibujar y las coordenadas que determinan el lugar.

`lineRGBA` no es la excepción: recibe un `renderer` y dos pares de `x` e `y` para determinar el principio y fin de la línea.

```
lineRGBA(renderer, 0, 0, WIDTH, HEIGHT, 0xff, 0x00, 0x00, 0xff);
lineRGBA(renderer, WIDTH, 0, 0, HEIGHT, 0x00, 0xff, 0x00, 0xff);
lineRGBA(
  renderer, 0, HEIGHT / 2, WIDTH, HEIGHT / 2, 0x00, 0x00, 0xff, 0xff);
```

Suponiendo que la pantalla tiene dimensiones `WIDTH`, `HEIGHT`, el código de arriba debe leerse como:

- una línea desde la esquina superior izquierda (`0`, `0`) hacia la esquina inferior derecha (`WIDTH`, `HEIGHT`).
- otra línea que va en la diagonal opuesta (desde la esquina superior derecha (`WIDTH`, `0`) hacia la esquina inferior izquierda (`0`, `HEIGHT`)).
- otra línea más que cruza horizontalmente la pantalla a la mitad de su altura (`HEIGHT/2`)

Como lo deduciras del nombre `lineRGBA` recibe 4 valores adicionales que representan el color de la línea: *red, green, blue* y *alpha*.

El color *alpha* determina que tan translúcido es el color:

- `0x00` para objetos totalmente transparentes.
- `0xff` para objetos totalmente opacos.

En este ejemplo hay 2 cuadrados donde el segundo es $\approx 40\%$ translúcido. Puede verse como su color se mezcla con el color que hay detras.

```
boxRGBA(renderer, 0, 0, 100, 100, 0x00, 0xff, 0x00, 0xff);
boxRGBA(renderer, 50, 50, 150, 150, 0xff, 0x00, 0x00, 0x60);
```

Muchas de las primitivas en `SDL2_gfx` permiten dibujar un contorno o una figura completa (aka *rellena*).

Aca hay dos ejemplos de como dibujar solo el contorno de un círculo (`circleRGBA`) y de como dibujar el círculo completo (`filledCircleRGBA`).

```
circleRGBA(
  renderer, WIDTH / 2, HEIGHT / 2, MIN / 2, 0x00, 0xff, 0x00, 0xff);
filledCircleRGBA(
  renderer, WIDTH / 2, HEIGHT / 2, MIN / 2, 0x00, 0xff, 0x00, 0xff);
```

Y hablando de círculos, con `filledEllipseRGBA` dibujamos elipses

[1] https://www.ferzkopp.net/Software/SDL2_gfx/Docs/html/index.html

```
filledEllipseRGBA(renderer,
                  WIDTH / 2,
                  HEIGHT / 2,
                  WIDTH / 2,
                  HEIGHT / 4,
                  0xff,
                  0x00,
                  0x00,
                  0xff);
```

Y polígonos? `SDL2_gfx` tiene `polygonRGBA` para dibujar polígonos arbitrarios como un triángulo o un rombo.

El único detalle es que hay que pasarle las coordenadas `x` e `y` de los vertices por dos vectores.

```
const short rhombus_x[] = { WIDTH / 2, WIDTH, WIDTH / 2, 0 };
const short rhombus_y[] = { 0, HEIGHT / 2, HEIGHT, HEIGHT / 2 };
assert(sizeof(rhombus_x) == sizeof(rhombus_y));
const int rhombus_n_points = sizeof(rhombus_x) / sizeof(rhombus_x[0]);

polygonRGBA(renderer,
            rhombus_x,
            rhombus_y,
            rhombus_n_points,
            0xff,
            0x00,
            0x00,
            0xff);
```

```
const short triangle_x[] = { WIDTH / 2, WIDTH, 0 };
const short triangle_y[] = {
  0, HEIGHT - 1, HEIGHT - 1
}; // offset a little to see the line
assert(sizeof(triangle_x) == sizeof(triangle_y));
const int triangle_n_points = sizeof(triangle_x) / sizeof(rhombus_x[0]);

polygonRGBA(renderer,
            triangle_x,
            triangle_y,
            triangle_n_points,
            0xff,
            0x00,
            0x00,
            0xff);
```

Código completo

```c
// Esta es una serie de mini-ejemplos de como usar SDL2_gfx,
// una libreria para SDL2 que permite dibujar ciertas primitivas
// en pantalla de maner muy simple.
//
// Tener en cuenta que este codigo *NO* hace un buen game loop
// ya que tiene un sleep (SDL_Delay) fijo pero para el proposito
// de mostrar estos mini-ejemplos alcanza.
//
// Compilar con:
// gcc `sdl2-config --cflags` -std=c11 gfx.c -lSDL2 -lSDL2_gfx -o gfx
//
// Ejecutar como:
// gfx example_number

#include <SDL2/SDL.h>
#include <SDL2/SDL2_gfxPrimitives.h>
#include <assert.h>
#include <stdio.h>

#define WIDTH 640
#define HEIGHT 480
#define MIN ((HEIGHT < WIDTH) ? HEIGHT : WIDTH)

#define N 7

int
main(int argc, char* argv[])
{
  int ret = -1;

  if (argc != 2) {
    fprintf(
      stderr, "Unexpected argument count.\nUsage:\n  %s [1-%d]\n", argv[0], N);
    return ret;
  }

  char* endptr = NULL;
  long int example_number = strtol(argv[1], &endptr, 0);
  if (!(0 < example_number && example_number <= N) || *endptr != '\0') {
    fprintf(stderr, "Unexpected example number.\n");
    return ret;
  }

  if (SDL_Init(SDL_INIT_VIDEO)) {
    fprintf(stderr, "SDL_Init failed: %s\n", SDL_GetError());
    return ret;
  }
```

```
SDL_Window* window =
  SDL_CreateWindow("SDL2 GFX", 100, 100, WIDTH, HEIGHT, SDL_WINDOW_OPENGL);
if (!window) {
  fprintf(stderr, "SDL_CreateWindow failed: %s\n", SDL_GetError());
  SDL_Quit();
  return ret;
}

SDL_Renderer* renderer = SDL_CreateRenderer(
  window, -1, SDL_RENDERER_ACCELERATED | SDL_RENDERER_PRESENTVSYNC);
if (!renderer) {
  fprintf(stderr, "SDL_CreateRenderer failed: %s\n", SDL_GetError());
  SDL_DestroyWindow(window);
  SDL_Quit();
  return ret;
}

SDL_Event e;
int begin = 1;

int quit = 0;
while (!quit) {
  if (SDL_PollEvent(&e)) {
    if (e.type == SDL_QUIT)
      quit = 1;
  }
  SDL_SetRenderDrawColor(renderer, 0, 0, 0, 0);
  SDL_RenderClear(renderer);


  const short rhombus_x[] = { WIDTH / 2, WIDTH, WIDTH / 2, 0 };
  const short rhombus_y[] = { 0, HEIGHT / 2, HEIGHT, HEIGHT / 2 };
  assert(sizeof(rhombus_x) == sizeof(rhombus_y));
  const int rhombus_n_points = sizeof(rhombus_x) / sizeof(rhombus_x[0]);


  const short triangle_x[] = { WIDTH / 2, WIDTH, 0 };
  const short triangle_y[] = {
    0, HEIGHT - 1, HEIGHT - 1
  }; // offset a little to see the line
  assert(sizeof(triangle_x) == sizeof(triangle_y));
  const int triangle_n_points = sizeof(triangle_x) / sizeof(rhombus_x[0]);

  switch (example_number) {
    case 1:
      if (begin)
        printf("Ejemplo de como dibujar lineas\n");

      lineRGBA(renderer, 0, 0, WIDTH, HEIGHT, 0xff, 0x00, 0x00, 0xff);
      lineRGBA(renderer, WIDTH, 0, 0, HEIGHT, 0x00, 0xff, 0x00, 0xff);
```

```
  lineRGBA(
    renderer, 0, HEIGHT / 2, WIDTH, HEIGHT / 2, 0x00, 0x00, 0xff, 0xff);
  break;
case 2:
  if (begin)
    printf("Ejemplo de como dibujar un circulo\n");

  circleRGBA(
    renderer, WIDTH / 2, HEIGHT / 2, MIN / 2, 0x00, 0xff, 0x00, 0xff);
  break;
case 3:
  if (begin)
    printf("Ejemplo de como dibujar un circulo y rellenarlo\n");

  filledCircleRGBA(
    renderer, WIDTH / 2, HEIGHT / 2, MIN / 2, 0x00, 0xff, 0x00, 0xff);
  break;
case 4:
  if (begin)
    printf("Ejemplo de como dibujar un rectangulo y rellenarlo "
           "traslucido\n");

  boxRGBA(renderer, 0, 0, 100, 100, 0x00, 0xff, 0x00, 0xff);
  boxRGBA(renderer, 50, 50, 150, 150, 0xff, 0x00, 0x00, 0x60);
  break;
case 5:
  if (begin)
    printf("Ejemplo de como dibujar un rombo\n");

  polygonRGBA(renderer,
              rhombus_x,
              rhombus_y,
              rhombus_n_points,
              0xff,
              0x00,
              0x00,
              0xff);
  break;
case 6:
  if (begin)
    printf("Ejemplo de como dibujar un triangulo\n");

  polygonRGBA(renderer,
              triangle_x,
              triangle_y,
              triangle_n_points,
              0xff,
              0x00,
              0x00,
              0xff);
```

```
      break;
    case 7:
      if (begin)
        printf("Ejemplo de como dibujar una elipse y rellenarla\n");

      filledEllipseRGBA(renderer,
                        WIDTH / 2,
                        HEIGHT / 2,
                        WIDTH / 2,
                        HEIGHT / 4,
                        0xff,
                        0x00,
                        0x00,
                        0xff);
      break;
    default:
      fprintf(stderr, "Unexpected example number.\n");
      SDL_DestroyRenderer(renderer);
      SDL_DestroyWindow(window);
      SDL_Quit();
      return ret;
  }

  SDL_RenderPresent(renderer);
  SDL_Delay(10);
  begin = 0;
 }

 ret = 0;

 SDL_DestroyRenderer(renderer);
 SDL_DestroyWindow(window);
 SDL_Quit();
 return ret;
}
```